DOUBLY LINKED LIST CODE:

#include<stdio.h>

#include<malloc.h>

typedef struct node

{

struct node \*prev;

int data;

struct node \*next;

}NODE;

typedef NODE\* NODEPTR;

NODEPTR start,end,tptr,newNode;

void insert(int gd)

{

newNode=(NODE\*)malloc(sizeof(NODE));

newNode->data=gd;

newNode->prev=NULL;

newNode->next=NULL;

if(start==NULL)

{

start=newNode;

end=newNode;

}

else

{

for(tptr=start;tptr&&tptr->data<gd;tptr=tptr->next);

if(tptr==NULL)

{

newNode->next=NULL;

end->next=newNode;//Note here it find end automatically

newNode->prev=end;//this end eliminate shadow pointer

end=newNode;

}

else if(tptr==start)

{

newNode->prev=NULL;

newNode->next=tptr;

tptr->prev=newNode;

start=newNode;

}

else

{

newNode->next=tptr->prev->next;

newNode->prev=tptr->prev;

tptr->prev->next=newNode;

tptr->prev=newNode;

}

}

}

void deleteNode(int givenData)

{

for(tptr=start;tptr && tptr->data!=givenData;tptr=tptr->next);

if(tptr==NULL)

printf("element not found");

else if(tptr==start)

{

(tptr->next)->prev=NULL;

start=tptr->next;

}

else if(tptr==end)

{

(tptr->prev)->next=NULL;

end=tptr->prev;

}

else

{

(tptr->prev)->next=tptr->next;

(tptr->next)->prev=tptr->prev;

}

}

void display()

{

for(tptr=start;tptr;tptr=tptr->next)

{

printf("%d",tptr->data);

}

}

int main()

{

int num,num2;

do

{

scanf("%d",&num);

if(num==-1)

{

break;

}

else

{

insert(num);

}

}while(1);

printf("\nAfter Insertion:\n");

display();

printf("\nEnter a node to delete:\n");

scanf("%d",&num2);

deleteNode(num2);

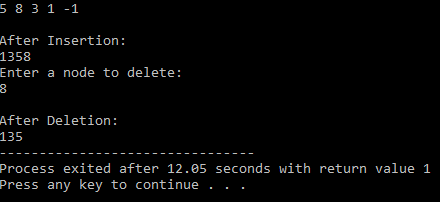
printf("\nAfter Deletion:\n");

display();

return 1;

}

**OUTPUT:**

****

**CIRCULARLY LINKED LIST CODE:**

#include<stdio.h>

#include<malloc.h>

typedef struct node

{

int data;

struct node \*next;

}NODE;

typedef NODE\* NODEPTR;

NODEPTR start,newNode,tptr,shadow,tptr2;

void insert(int givenData)

{

newNode=(NODEPTR)malloc(sizeof(NODEPTR\*));

newNode->data=givenData;

newNode->next=NULL;

if(start==NULL)

{

start=newNode;

newNode->next=newNode;

}

else

{

for(tptr=start,shadow=NULL;(tptr->next!=start)&&(tptr->data<givenData);shadow=tptr,tptr=tptr->next);

if(tptr==start && tptr->data>givenData)

{

newNode->next=tptr;

for(tptr=start;tptr->next!=start&&tptr;tptr=tptr->next);

tptr->next=newNode;

start=newNode;

}

else if(tptr->next==start && tptr->data<givenData)

{

tptr->next=newNode;

newNode->next=start;

}

else

{

shadow->next=newNode;

newNode->next=tptr;

}

}

}

void delete(int givenData)

{

for(tptr=start,shadow=NULL;(tptr->data!=givenData)&&(tptr->next!=start);shadow=tptr,tptr=tptr->next);

{

if(tptr==start)

{

for(tptr2=start;tptr2->next!=start;tptr2=tptr2->next);

tptr2->next=tptr->next;

start=tptr->next;

}

else if(tptr->data!=givenData)

{

printf("Element Not found");

}

else

{

shadow->next=tptr->next;

}

}

}

void display()

{

NODEPTR print;

print = start;

do{

printf("\n%d\n",print->data);

print=print->next;

}while(print!=start);

}

int main()

{

int num,num2;

do

{

scanf("%d",&num);

if(num==-1)

{

break;

}

else

{

insert(num);

}

}while(1);

display();

printf("\nEnter a number to delete:\n");

scanf("%d",&num2);

delete(num2);

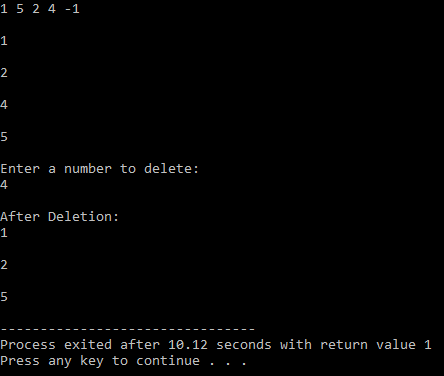
printf("\nAfter Deletion:");

display();

return 1;

}

**OUTPUT:**

****